Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 186 – 195

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

An Evaluation of Molecular Dynamics Performance on the Hybrid Cray XK6 Supercomputer

W. Michael Brown[a,*], Trung D. Nguyen[a], Miguel Fuentes-Cabrera[b], Jason D. Fowlkes[b], Philip D. Rack[c], Mark Berger[d], Arthur S. Bland[a]

[a]*National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN, USA*
[b]*Center for Nanophase Materials Sciences and Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN*
[c]*Materials Science and Engineering Department, The University of Tennessee, Knoxville, TN, USA*
[d]*NVIDIA, Santa Clara, CA, USA*

Abstract

For many years, the drive towards computational physics studies that match the size and time-scales of experiment has been fueled by increases in processor and interconnect performance that could be exploited with relatively little modification to existing codes. Engineering and electrical power constraints have disrupted this trend, requiring more drastic changes to both hardware and software solutions. Here, we present details of the Cray XK6 architecture that achieves increased performance with the use of GPU accelerators. We review software development efforts in the LAMMPS molecular dynamics package that have been implemented in order to utilize hybrid high performance computers. We present benchmark results for solid-state, biological, and mesoscopic systems and discuss some challenges for utilizing hybrid systems. We present some early work in improving application performance on the XK6 and performance results for the simulation of liquid copper nanostructures with the embedded atom method.

Keywords: Molecular Dynamics, Materials, GPU, Supercomputer, Embedded Atom Method

1. Introduction

The trend for increasing computational performance through increased CPU frequencies has been hindered due to issues with power consumption, heat dissipation, and high memory access latencies, among other things. These issues have resulted in new strategies for performance improvements that typically require increasing software parallelism with careful management of data locality. The concept of heterogeneous architectures with specialized compute units for performing different tasks has become a popular idea for increasing parallelism with electrical power and cost efficiency. Basic examples include hybrid systems that combine a traditional CPU with a coprocessor or accelerator such as a graphics processing unit (GPU), digital signal processor, or a field-programmable gate array. Due to dramatic improvements in the hardware and software available for programming GPUs for general purpose computing, with NVIDIA's Compute Unified Device Architecture (CUDA) [1] being the most notable example, hybrid computers with GPU accelerators have become popular in high performance computing. Already, a number of the highest performing supercomputers in the Top500 list [2] use GPUs.

*Corresponding author: Mike Brown *Email address:* brownw@ornl.gov

doi:10.1016/j.procs.2012.04.020

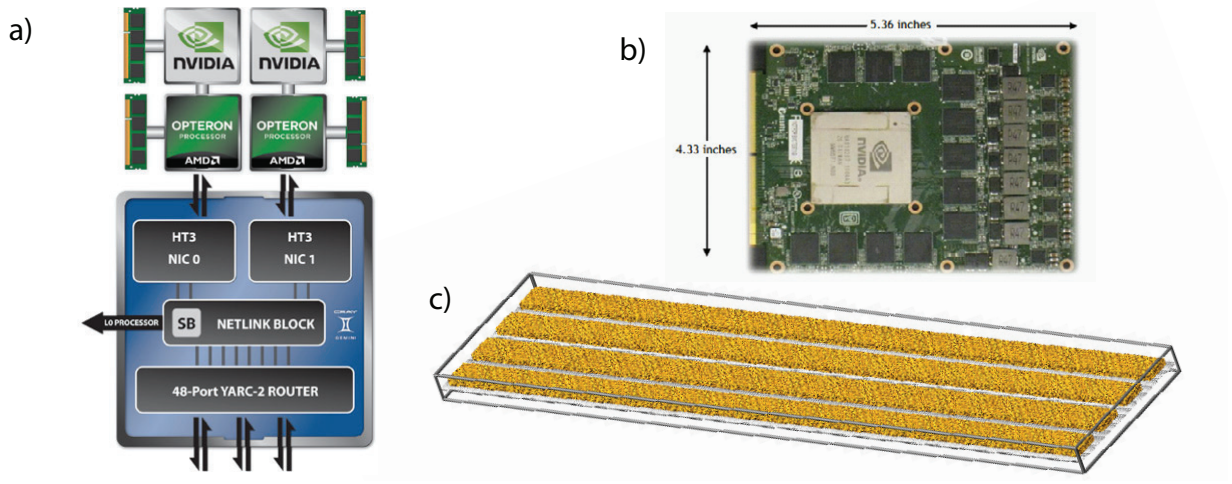

Figure 1: a) Cray XK6 Architecture b) Tesla X2090 (without heat sink) c) Initial configuration for copper lines deposited on a graphitic substrate.

The Cray XK6 system has recently been released as a hybrid supercomputer capable of scaling up to 50 petaflops with the use of NVIDIA GPUs. In this paper, we describe the Cray XK6 architecture and the deployment at Oak Ridge National Laboratory that will be named Titan. We provide an early performance evaluation of molecular dynamics (MD) performance on the XK6 using the LAMMPS package [3]. We note that a number of different MD codes exist that have been ported or written for GPU acceleration that are highly optimized for specific applications [4]. Here, we have restricted our focus to LAMMPS due to its general-purpose capabilities for simulating a wide variety of systems that are of interest in multiple scientific disciplines. Because LAMMPS has been developed as an open source modular package with contributions from many developers at many different institutions, the experiences and challenges in porting the code for use on next generation architectures are relevant to other software efforts involving object-oriented legacy codes in computational physics.

The algorithms we have developed for efficient GPU acceleration in LAMMPS have been published in detail [5, 6]. Here, we present benchmark results for solid-state, biological, and mesoscopic systems on the XK6. We present benchmarks results for CPU-based simulations on the XK6 for up to 8192 nodes with a performance comparison to the previous XT5 architecture. We present results for GPU-accelerated simulations on a development partition with NVIDIA Tesla X2090 GPUs. We describe early work to improve performance on the XK6. We present a implementation of the embedded atom method (EAM) [7] using GPU acceleration along with performance results for simulation of copper lines deposited on a graphitic substrate and discuss future work.

2. Hardware and Methods

2.1. XK6 Architecture and the Tesla X2090 Accelerator

The Cray XK6 cabinet can house up to 96 compute nodes, each using a 16-core 64-bit AMD Opteron 6200 series processor and 16 or 32 GB registered ECC DDR3 SDRAM. The Gemini ASIC is connected directly to 2 Opteron processors (Figure 1a) through HyperTransport 3, bypassing the PCI links to result in a sub-microsecond latency for remote puts, 1-2 microsecond latency for other point-to-point messages, and 20 GB/s of injection bandwidth per node. As with previous Cray interconnects, Gemini is connected in a 3-D torus topology to improve bisection and global bandwidth characteristics and it supports dynamic routing of messages. Each Gemini chip has 48 switch ports (160 GB/s internal switching capacity per chip), full ECC protection, and adaptive routing to mask out lane failures. Each compute node can house a Tesla X2090 accelerator (connected via PCE-e 2.0) to create a hybrid system with a peak performance exceeding 70 Tflops per cabinet.

The NVIDIA Tesla series is designed specifically for high performance computing. Tesla processors are efficient, massively parallel processors that excel at tackling large amounts of similar data, by splitting tasks into hundreds or

thousands of pieces to be calculated simultaneously. The NVIDIA Tesla X2090 GPU Computing Processor Module, currently used in the XK6, is a custom form factor SXM (Server PCI Express Module) based on the current-generation NVIDIA Fermi architecture. It comprises a computing subsystem with a GPU and 6 GB of GDDR5 on-board memory clocked at 1.85 GHz. The SXM form factor requires no external connectors as unused MXM signals are converted to additional power and grounds in order to accommodate the additional power requirements. The X2090 module, Figure 1b, is shorter in length compared to a standard PCI-e form factor module such as the Tesla M2090, and is ideal for use in systems in which the standard form does not fit because of space limitations or custom cooling requirements. The X2090 specs are similar to the M2090, with 512 processor cores @ 1.3 GHz and 225 W power dissipation.

2.2. XK6 Deployment at ORNL

The XK6 system is being deployed at ORNL by upgrading the existing Jaguar [8] system in 2012 to the Cray XK6 compute nodes and Gemini interconnect. The first phase of the upgrade involved replacement of all 18,688 XT5 compute nodes and 256 Service and I/O nodes with 18,688 XK6 compute nodes and 512 XIO nodes. The effect of this was to double the system memory from 300 terabytes to 600 terabytes, increase the number of AMD Opteron processor cores from 224,256 to 299,008, and replace the 2005 vintage SeaStar interconnect with Crays latest Gemini interconnect. In the first phase, 960 of the compute nodes include NVIDIA Tesla X2090 GPUs to provide a test platform to allow our users to begin porting applications to the GPU platform. This phase was completed in the first quarter of 2012.

The second phase of the upgrade will replace the 960 NVIDIA X2090 GPUs with between 10 and 20 petaflops of NVIDIAs next generation Kepler GPUs. The details of the Kepler GPU are proprietary at this writing, but the new processor is expected to be substantially faster than the current generation X2090 with more than one teraflop of double precision performance per chip. The second phase is expected to be completed in the first quarter of 2013, and when completed the system will be renamed from Jaguar to Titan.

2.3. LAMMPS

LAMMPS supports GPU acceleration for short-range force calculation [5] with optional acceleration for neighbor list builds and/or particle-particle particle-mesh (P^3M) long-range electrostatics [6]. Neighbor list builds are performed on the accelerator by first constructing a cell list that is utilized to build a verlet list using a radix sort to assert deterministic results. The van der Waals and short-range electrostatic forces are computed in a separate kernel. For each particle, the force-accumulation is performed by one or multiple threads. A default number of threads is chosen based on the hardware and the potential model being used for calculation. Force models currently supported for GPU acceleration include several variants of the Lennard-Jones potential with and without coulombic interactions, the Morse potential, the Buckingham potential, the CHARMM potential, tabulated potentials, the coarse-grain SDK potential, and the anisotropic Gay-Berne and RE-squared potentials. Combinations of these potentials and also potential models that have not been ported for GPU acceleration can be used by specifying "hybrid" models. Three precision modes are supported for GPU acceleration - single, mixed, and double. In mixed precision mode, all particle data is stored in single precision and most computations are performed in single precision. The exception is accumulation, which is performed in double precision for all values. Forces, energies, and virials are stored in double precision when using this mode. For long-range electrostatics, GPU acceleration for P^3M is supported for charge assignment to the mesh and force interpolation. The parallel FFT is performed on the host. The P^3M calculation can be performed in single or double precision.

All of the statistics computations, thermostats, barostats, time integration, bond/angle/dihedral/improper calculations, and any other simulation modifications are performed on the host. In order to achieve efficient GPU acceleration, these calculations must be parallelized within each node on the host [5]. This is performed by using multiple MPI processes, each sharing one or more GPUs on a compute node. This approach offers several options for host/GPU concurrency. First, short-range force calculation can be performed simultaneously on the host and the GPU with dynamic load balancing [5]. Second, short-range force calculation and optionally P^3M calculation can be performed on the GPU simultaneously with bond/angle/dihedral/improper calculations and optionally long-range electrostatics calculations on the host. Third, hybrid models using multiple potential models for force calculation support simultaneous calculation of specified models on the GPU with other models calculated on the host.

In this work, we provide benchmark results on the XK6 with evaluation of several improvements recently made to LAMMPS to increase performance. In order to achieve good parallel efficiency, it is desirable to obtain efficient

acceleration with low particle counts on each process. In general, increasing the number of threads performing force accumulation for a particle can improve performance at small particle counts. The trade-off is an increase in the number of calculations required and different memory access patterns for particle data. In our past work, we have used a single neighbor list storage format with a dense row-major matrix where each column contains the neighbor indices for one particle. The problem with this approach is that it only results in contiguous memory access for neighbors when the number of threads per particle is 1. Therefore, we have modified the neighbor storage to result in contiguous memory access for an arbitrary number of threads assigned to each particle. This increases the time required to perform neighbor list builds on the GPU, but is generally a win for the overall short-range calculation - especially for models requiring large neighbor lists. For an atomic Lennard-Jones fluid with a cutoff of 2.5σ, we can efficiently use four times the number of threads per particle for force calculation on the GF100 series GPU chips.

The XK6 uses Interlagos CPUs with 16 cores on a socket. Although LAMMPS does have some support for on-node OpenMP parallelism to reduce the number of processes running on a node, at writing, it is still more efficient to run using only the MPI spatial decomposition for many cases. It is therefore important to optimize the process-to-grid mapping in order to minimize the amount of off-node MPI communications. Although the MPI standard provides routines for obtaining process mappings for Cartesian grids (e.g. the MPI_Cart_create routine), at writing, neither the default MPICH nor OpenMPI implementations take multicore CPUs into account when mapping processes. We therefore implemented a two-level grid factorization into LAMMPS in order to reduce off-node MPI communications. In this case, the grid dimensions on each node are obtained using the factorization of the MPI processes on a node that minimizes the surface area calculated with the box length ratios of the entire simulation domain. These dimensions are then used to divide the simulation box into subdomains for each node, again with a factorization that minimizes subdomain surface area. The neighboring *nodes* for ghost exchanges are obtained using an MPI Cartesian communicator with a single process per node. Finally, the factorization is performed again on each node, but using the actual subdomain box lengths in the calculation. From this result, the neighbors for each MPI process are obtained with the intent of maximizing the mean number of neighbors for each MPI process that are on the same node.

The embedded atom method (EAM) [7] is a common model used for simulation of metals and metal alloys, and therefore we have implemented a capability for GPU acceleration into this model within LAMMPS. The GPU acceleration supports the original EAM potential as well as a generalized form for metals and alloys presented by Finnis and Sinclair [9]. For an atom i, the energy given by the standard EAM potential is,

$$E_i = F_\alpha\left(\sum_{j\neq i}\rho_\beta(r_{ij})\right) + \frac{1}{2}\sum_{j\neq i}\phi_{\alpha\beta}(r_{ij}), \tag{1}$$

where F is the embedding energy calculated from the atomic electron density ρ, ϕ is a pair potential interaction, and α and β are the element types of atoms i and j. The model is effectively a many-body potential because the electron charge density at each neighboring atom position must be calculated with a loop over surrounding atoms within some cutoff. Therefore, GPU acceleration is achieved with two kernels in order to allow for calculation and ghost exchange of electron density followed by calculation of the energies, forces, and virial terms for each atom. The requirement for ghost atom data exchange during force calculation is a unique requirement of the EAM model as opposed to the other models considered here, and this limits concurrency options due to the additional synchronization required.

For simulations that require consideration of long-range electrostatics, particle-mesh methods are commonly used to achieve a favorable time complexity with the use of fast Fourier transformations (FFTs). Solving 3D FFTs in parallel is often the bottleneck for achieving parallel efficiency for molecular simulations due to the effectively all-to-all MPI communications required. Because components of the short-range real space calculations and long-range reciprocal space electrostatic calculations can be performed independently, MD codes can often improve parallel efficiency by assigning the long-range reciprocal-space calculations requiring FFTs to a subset of MPI processes [10]. This allows for simultaneous calculation of short-range and long-range components and can be used to reduce the number of MPI processes involved in relatively small parallel FFT calculations. This feature has recently been added to LAMMPS by Yuxing Peng and Chris Knight from the Voth Group at the University of Chicago. We have adapted the GPU-acceleration library to work with this approach and here we evaluate the impact on performance.

We provide performance results for LAMMPS with these new improvements on the XK6 for a variety of benchmarks as described in the results section. For these tests, the 2011-8-22 version of LAMMPS was used with some updates and unreleased enhancements as described above. Host code was compiled using the Cray GNU program-

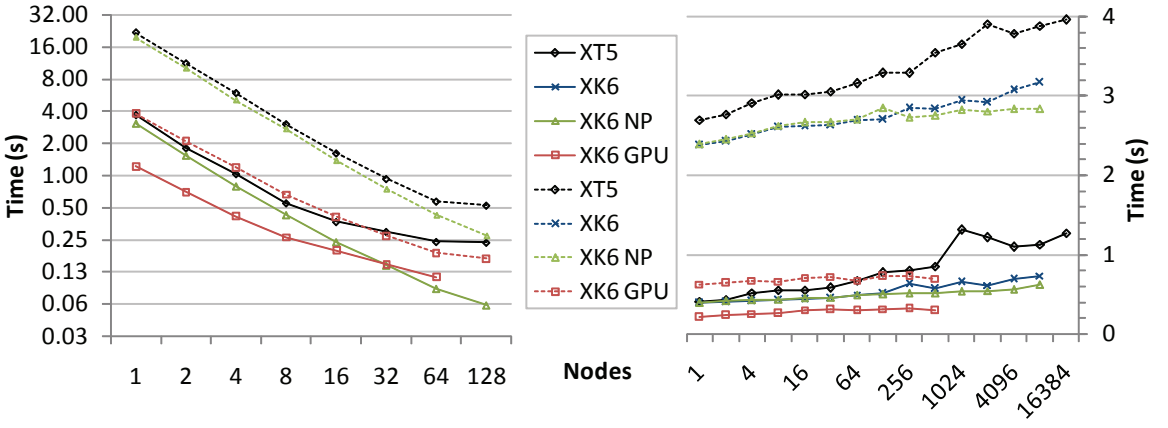

Figure 2: Simulation results for an atomic fluid with the Lennard-Jones potential for a cutoff of 2.5σ (solid lines) and 5.0σ (dashed lines). "XK6 NP" results use the two-level factorization for process mapping as do the GPU results. Left: Strong scaling for a fixed-size simulation of 256K atoms. Right: Simulation times from weak scaling with 32K atoms per node. "XK6 GPU" was run using 4 processes per node (PPN) for weak scaling; PPN for strong scaling was chosen based on the node count. Runs without acceleration used all available cores.

ming environment 4.0.30 with the GNU 4.6.2 C++ compiler, MPT 5.4.0, and the Cray optimized FFTW 3.3.0.0 library. Host code was compiled with "-O2 -march=amdfam10 -ftree-vectorize" optimization. Device driver version was 270.41.32. Device code using CUDA was compiled with version 4.0.17 of the CUDA toolkit. A static executable was generated for use in all tests. For tests run on XT5 hardware, the XT5 partition of Jaguar was used (18,688 compute nodes with dual 2.6GHz hex-core AMD Opteron 2435 processors and 16GB of DDR2-800 memory per node). For tests on XK6 hardware, a 96 cabinet upgrade partition for Jaguar was used. For tests with GPU acceleration, 10 XK6 development cabinets with 960 Tesla X2090 GPUs (ECC on) were used. All GPU accelerated simulations were performed using mixed precision (with double precision P^3M) as opposed to double precision for other simulations.

3. Results

For the first benchmark, we have used an atomic fluid in the microcanonical ensemble simulated with the Lennard-Jones potential for van der Waals forces. Strong scaling tests were performed with 256K atoms and weak scaling tests were performed using 32K atoms per node. The reduced density for the liquid is 0.8442. Simulations were performed using force cutoffs of 2.5σ and 5.0σ with a neighbor skin of 0.3σ. For comparison, we also performed simulations on Jaguar with the previous XT5 hardware (dual Istanbul/SeaStar 2+ interconnect). Results are shown in Figure 2.

For a 2.5σ cutoff on a single node, 256K atom simulations were 17% faster on the XK6 versus the XT5 and 32K atom simulations were 4% faster (XT5 times were 3.685s and 0.402s). With GPU acceleration, the simulation rate was 3.03 times faster than the XT5 for 256K atoms and 1.92 times faster for 32K atoms. In addition to atom count, the number of neighbors per atom for short-range forces will have a significant impact on relative GPU performance. This is because 1) the short-range force calculation offers the highest potential for speedup with GPU acceleration versus other routines and the percent time spent in short-range calculations will increase with cutoff and 2) the force accumulation can be divided between more GPU threads when the cutoff is higher. For a 5.0σ cutoff, for example, the speedup versus the XT5 for 256K atoms is 5.68 (XT5 time: 21.657s). For 32K atoms, the XK6 with GPU acceleration is 4.33 times faster than the XT5 (XT5 time: 2.691s).

In general, using GPU acceleration on the XK6 will decrease parallel efficiency for two reasons. First, the GPU is designed for massive data parallelism and will often run more efficiently with many more threads in flight than available processing cores. This allows for amortization of the time for memory access through simultaneous computations from other threads. Decreasing the number of threads run on a GPU per timestep will often decrease GPU performance relative to the CPU as shown in the strong scaling tests in Figure 2. For the 2.5σ cutoff, it is faster to run without GPU acceleration below 8000 atoms per node.

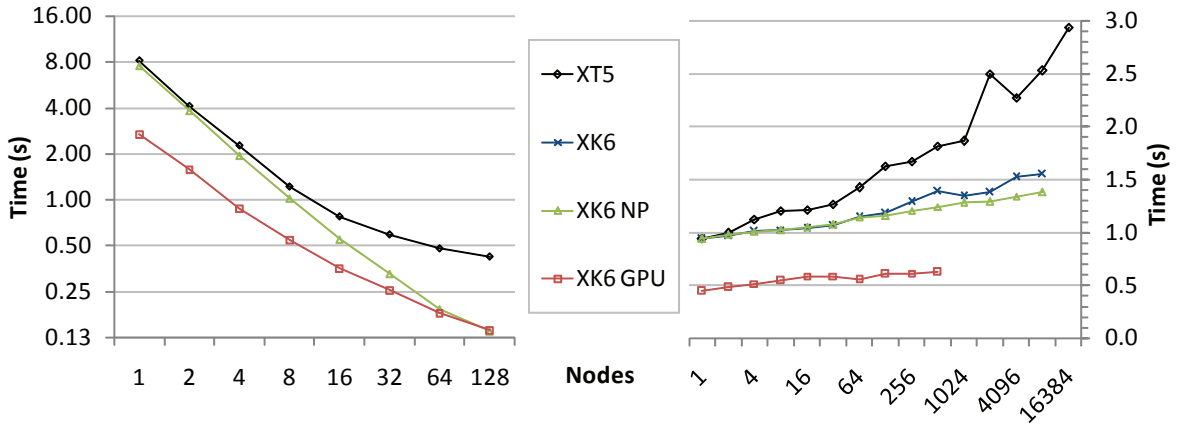

Figure 3: Simulation results for the copper metallic solid benchmark with the embedded atom method. "XK6 NP" results use the two-level factorization for process mapping as do the GPU results. Left: Strong scaling for a fixed-size simulation of 256K atoms. Right: Weak scaling with 32K atoms per node. "XK6 GPU" was run using 4 processes per node (PPN) for weak scaling; PPN for strong scaling was chosen based on the node count. Runs without acceleration used all available cores.

The second reason is obvious - because GPU accelerated runs are significantly faster, MPI communications and noise will constitute a larger percentage of the run time. When comparing to the XT5, the dramatic improvements in the Gemini interconnect will be necessary in order to run with GPU acceleration efficiently at scale (Figure 2). Despite the similar timings for the atomic fluid on a single node for the XT5 and XK6 for scaled-size tests, large parallel runs were up to twice as fast on the XK6 due to reduced communication times. Software improvements to further reduce the impact from communications and noise will always be desirable. Because the XK6 has 16 cores on a single socket, the MPI process mapping can have a significant impact on performance. Using the two-level factorization for mapping described above, performance on the XK6 can be improved by an additional 20% for large runs without acceleration as shown in Figure 2 ("XK6" vs "XK6 NP"). Although parallel efficiency when using acceleration is similar to the XT5 for weak-scaling tests using 32 nodes or less, larger jobs have similar parallel efficiencies to the XK6 runs without acceleration. This is due, in part, to the ability to efficiently run with fewer MPI processes per node.

For the embedded atom method, we used a standard benchmark for simulation of a copper metallic solid in the microcanonical ensemble. Again, the strong scaling tests use a simulation size of 256K atoms and the weak scaling tests use 32K atoms per node. The force cutoff is 4.95Å with a neighbor skin of 1.0Å. Results are shown in Figure 3. On a single node, the XK6 CPU simulations are 7.3% faster for 256K atoms and require the same amount of time as the XT5 for 32K atoms (XT5 times: 8.173s, 0.938s). With acceleration, the simulations are 3.05 and 2.12 times faster than the XT5, respectively. Although the EAM potential requires approximately 2.4 times longer for force calculation than the Lennard-Jones model (2.5σ cutoff), the relative GPU speedups are not as substantial as might be expected. There are two reasons for this result. First, the EAM potential uses splines to calculate functions from tabulated values and this approach results in low arithmetic intensity. Second, electron densities must be transferred to the host for ghost atom exchange during force calculation and this requires additional synchronization and MPI communication.

For parallel performance, the XK6 simulations with Gemini showed substantial performance improvements with 3 times the simulation rate for 32K atoms on 128 nodes and 1.5 times the simulation rate for 32K atoms per node on 8192 nodes. With use of the two-level factorization for process mapping, performance is further improved on the XK6 by up to 12%. Relative performance at low atom counts is higher when compared to the atomic fluid benchmark in strong scaling tests and parallel efficiency with weak scaling is similar, despite the additional ghost atom data exchange required for EAM.

For molecular systems, we have used the standard rhodopsin benchmark packaged with LAMMPS. The benchmark simulates the all-atom rhodopsin protein in a solvated lipid bilayer with the CHARMM force field. Long range electrostatics are calculated with P^3M. The simulation is performed in the isothermal-isobaric ensemble with SHAKE constraints and a timestep of 2.0 femtoseconds. The model contains counter-ions and a reduced amount of water to

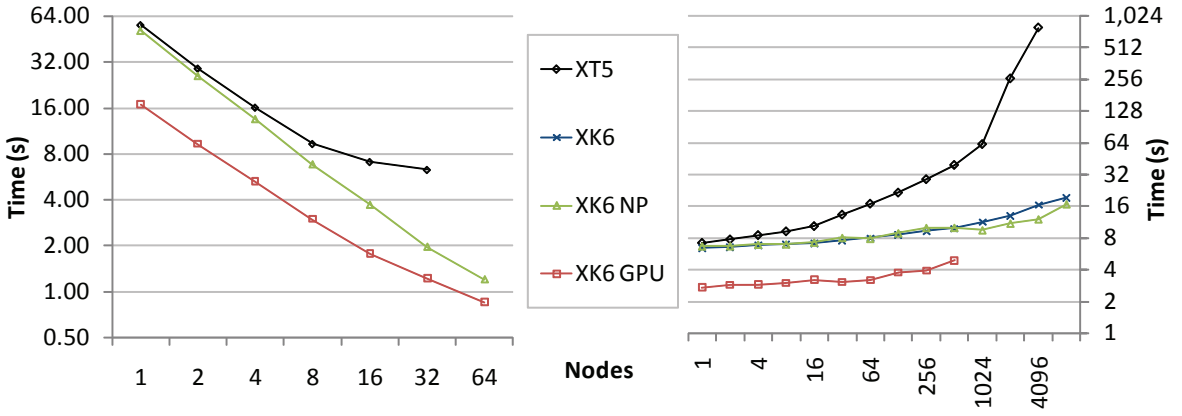

Figure 4: Simulation results for the rhodopsin benchmark. "XK6 NP" results use a separate partition of processors for the reciprocal space P^3M calculations (2 PPN) than used for the real space short range calculations (14 PPN). The GPU results also use separate partitions (8 PPN for weak scaling tests). Left: Strong scaling for a fixed-size simulation of 256K atoms. Right: Weak scaling with 32K atoms per node.

make a 32K atom system. An inner cutoff of 8Å and an outer cutoff of 10Å are used for short-range force calculations with a neighbor skin of 1.0Å. In order to show results for a larger simulation with similar properties, strong scaling benchmarks were performed using a 256000 atom initial configuration obtained from replicating the rhodopsin simulation box 8 times in order to double the length in each dimension. For weak scaling tests, the simulation box is replicated such that there are 32K atoms per node. Results for the benchmarks are shown in Figure 4.

When compared to the XT5, XK6 CPU runs show a 10.4% performance improvement on a single node for 256K atoms and a 10.3% performance improvement for 32K atoms (XT5 times: 55.718s and 7.106). The most significant performance improvement is shown for parallel simulations. In strong scaling tests, the XK6 simulations were 2.8 times faster on 32 nodes; on the XT5 scaling beyond 32 nodes resulted in worse performance. For weak scaling tests, relative Gemini performance was outstanding and the simulations on 4096 nodes were 47.6 times faster than the XT5. As discussed in the Methods section, assigning components of the long-range electrostatics calculation to a subset of MPI processes can be used to further improve performance. Using 1 MPI process per NUMA node (2 PPN) for reciprocal space P^3M calculations and 7 MPI processes per NUMA node (14 PPN) for real space short-range calculations improves the speedup at 4096 nodes to 65.4. Although performance below 1024 nodes was similar with use of this option, at 1024 nodes and higher, performance was improved by 14 to 27.2 percent on the XK6 with use of this option (Figure 4).

With GPU acceleration, the results in Figure 4 also use a separate partition for long range calculations. On a single node, XK6 results were 3.3 times faster than on the XT5 for 256K atoms and 2.6 times faster with 32K atoms. Performance for parallel simulations resulted in higher parallel efficiencies when compared to the atomic fluid and copper benchmarks due to the higher arithmetic intensity for the CHARMM short range calculation and the increase in the number of neighbors included in the force calculation for each atom. Both of these differences allow the force accumulation to be split between a larger number of GPU threads, allow for more MPI processes per node, and improve relative performance at smaller atom counts. For weak scaling tests, parallel efficiency was on par with results using only the XK6 CPU. Interestingly, performance when using a separate partition for the long range calculations was significantly higher for GPU-accelerated runs (data without separate partitions not shown). Although performance at 128K atoms per node and higher was not improved, there were significant improvements at lower atom counts, even when using only a single node. For the case with 32K atoms per node with 4 MPI processes for long range and 4 MPI processes for short range on each node, performance was improved by 10.5% to 21.1% in weak scaling tests.

For the last benchmark, we have used simulations based on our previous studies of liquid crystal mesogens [11] that result in high speedups with GPU acceleration due to the high arithmetic intensity [5]. In this case, the mesogens are represented with biaxial ellipsoid particles with semiaxes lengths of 2σ, 1.5σ, and 1σ. The ellipsoids are modeled with the Gay-Berne potential generalized for biaxial particles [12]. The particles are simulated in the isotropic phase

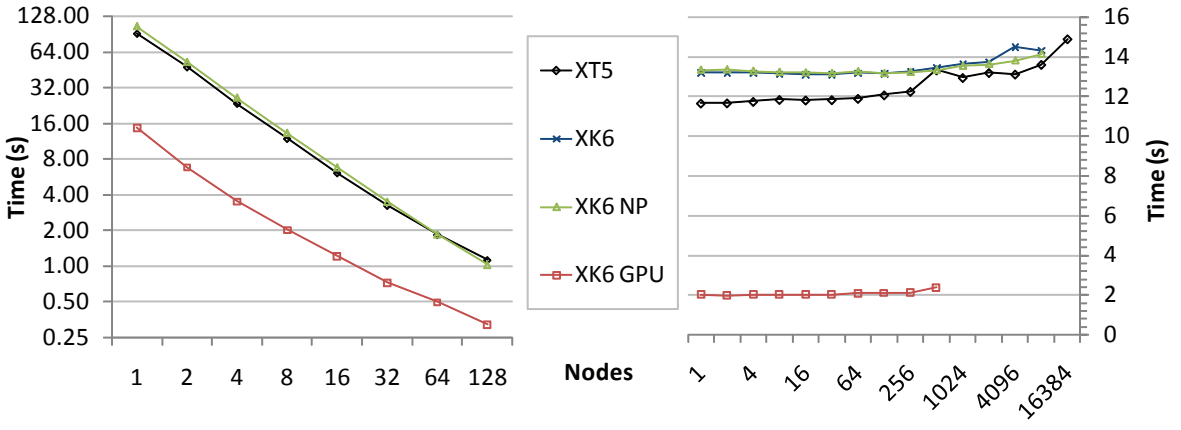

Figure 5: Simulation results for the liquid crystal benchmark. "XK6 NP" results use the two-level factorization for process mapping as do the GPU results. Left: Strong scaling for a fixed-size simulation of 262144 particles. Right: Weak scaling with 32768 particles per node. "XK6 GPU" was run using 10 PPN for weak scaling; PPN for strong scaling was chosen based on the node count. Runs without acceleration used all available cores.

and sampled from the isothermal-isobaric ensemble with a reduced temperature of 2.4 and a reduced pressure of 8.0. A cutoff of 4σ is used with a 0.8σ neighbor skin. Fixed-size simulations with 262144 particles were used for strong scaling tests and weak scaling tests were performed with 32768 particles per node. The results are shown in Figure 5.

On a single node, the results are slower on the XK6 when using only the CPU; XT5 performance was 11.6% - 11.7% faster for both simulations (XT5 times: 91.261s and 11.653s). With GPU acceleration, however, the simulations ran 6.23 times faster than the XT5 for 262144 particles and 5.82 times faster with 32768 particles. The high speedups are due to the very high arithmetic intensity for the anisotropic force calculations and the use of mixed precision with native GPU transcendentals. For the same obvious reasons, parallel efficiency is also very high when compared to the other benchmarks. Speedups with GPU acceleration were high for all strong scaling tests and parallel efficiency for the XT5, XK6, and XK6 with GPU acceleration was similar for the weak scaling tests.

We have also examined XK6 performance for simulations to investigate dewetting of metallic nanoparticles - a process that can be used to control self-assembly on a substrate. Dewetting can proceed via two processes, the Rayleigh-Plateau instability and the spinodal thin film instability. Both produce oscillations on the metallic film that lead to its breakage and eventual transport into small droplets. For example, a metallic thin line breaks up in small droplets, the distance between them given by the Rayleigh-Plateau instability wavelength of maximum growth. Similarly, a metallic thin film breaks in small droplets where distance is accounted for by the spinodal dewetting phenomenon (for further explanation and details, see [13]).

Using both experiment and theory we have investigated ways of controlling how the breakage occurs so as to control the assembly of metallic nanoparticles. The theoretical calculations are based on MD simulations and employ well-known potentials to mimic the atomic interactions [13, 14, 15]. Currently, we are investigating the dewetting of Cu-lines on graphite. Due to size effects, it is necessary to use high performance computers for simulation — simulation of smaller lines results in contraction before breakage can occur. Simulation with periodic boundaries results in breakage and reveals that the Rayleigh-Plateau instability happens also at the nanoscale. However, use of lines with infinite length misses edge effects in which a pearling effect creates droplets at the end of the line that are not the consequence of the Rayleigh instability phenomenon. Also, with periodic boundaries, we do not observe a bi-modal distribution of droplets, i.e., a distribution in which a large drop is associated with the wave peak and a small satellite droplet forms at the wave troughs during pinch-off.

The LAMMPS work described in this paper is also of benefit to these investigations, as we have modeled the Cu-Cu interactions with EAM and the Cu-C interactions with a Lennard-Jones potential fit so as to reproduce the binding energy of the Cu(111)/graphite interface and the contact angle of Cu on graphite. In order to determine performance gains in this work, we have modeled Cu-lines deposited on an immobile graphitic substrate. The lines are 5450Å in length, 200Å in width and 50Å in thickness. An initial configuration for a simulation of 4 Cu-lines deposited on

graphite is shown in Figure 1c. In this case, the spacing between adjacent lines is 100Å and the gap between the line ends to the box edges is 50Å so as to avoid artificial effects from periodic boundary conditions. The substrate consists of approximately 10M carbon atoms arranged into three layers, and the simulations are performed at 1500K. The force cutoff is 4.95Å for the EAM interaction between Cu atoms and 11.0Å for the Lennard-Jones interactions. When performed without GPU acceleration, there is no performance advantage for using the XK6 for simulation on 512 nodes. With acceleration, however, the simulation completes 2.7 times faster, allowing for simulation at a rate of 11 nanoseconds per day for 23.6M atoms with GPU acceleration. This result is on par with the bulk copper and atomic fluid benchmarks presented above - with approximately 46K atoms per node and a higher cutoff for Lennard-Jones interactions than the 2.5σ atomic fluid.

4. Discussion

For all of the simulations presented, we were able to achieve significant performance improvements on the XK6 hardware for Jaguar when compared to the previous generation XT5. Although the Interlagos chip puts 16 cores on a single socket with a lower clock rate, performance improvements were obtained for all but the liquid crystal benchmark when compared to dual-socket Istanbul nodes. The most significant performance improvements resulted from use of GPU acceleration and also from the Gemini interconnect for parallel simulations. The relative performance improvement depends on the model and the system being simulated. On a single node, the speedups versus the XT5 for approximately 32K particles were as follows: atomic fluid (2.5σ cutoff) - 1.92X, atomic fluid (5.0σ cutoff) - 4.33X, bulk copper - 2.12X, rhodopsin protein - 2.6X, and liquid crystal - 5.82X. For simulations using P^3M for long-range electrostatics calculations, Gemini performance is outstanding. Although limited GPUs prevented benchmarks at 1024 nodes or higher, performance with P^3M was already 8 times faster versus the XT5; CPU-only simulations on 4096 nodes were 65.4 times faster.

These results are very encouraging for early runs on the XK6 with Tesla X2090 GPUs. For Titan, the use of next-generation Kepler GPUs and NVIDIA drivers with improved support for sharing between multiple MPI processes will offer performance improvements for existing codes. Additionally, we expect significant performance improvements from code modifications to optimize for the XK6 architecture. Here, we have presented results with improvements to GPU global memory access patterns for neighbor lists, new algorithms for MPI process mapping for ghost exchange, and separate partitions for long-range calculations in LAMMPS. These modifications result in significant performance improvements for GPU computations and overall parallel efficiency. Additional software improvements to improve parallel efficiency will always be desireable. Our efforts are currently focused on methods to improve management of multiple MPI processes sharing a single GPU, overlapping host-device and MPI communications with GPU calculations, and further improvements to long-range electrostatics calculations. Overlapping communications and computation should be especially beneficial for EAM; interactions with local (non-ghost) atoms can be evaluated concurrently with communications for electron density exchanges. For long-range electrostatics, trivial optimizations such as adjustment of P^3M parameters (e.g. the cutoff used for real-space computations) for the XK6 will likely yield some performance gains. Efforts to improve existing approaches and implement/evaluate alternative algorithms for long-range electrostatics are already underway. These include using analytic gradient calculations for P^3M to halve the number of FFTs required [5] and multigrid or similar approaches for performing long-range calculations with improved parallel efficiency. Cutoff-based approaches for long-range electrostatics [16] are also an interesting consideration; these methods typically require many more neighbors per atom than used in the atomic fluid 5.0σ benchmark and therefore can be much more competitive with GPU acceleration.

With the hardware, driver, and software improvements planned for Titan, we expect to offer significant performance improvements on Titan when compared to the previous generation Jaguar XT5 hardware on a node-per-node basis. For some models and systems, the machine will allow investigators to realize time scales and system sizes not previously possible. Although relative GPU performance is generally better when running with larger particle counts on each GPU, for the system sizes of interest on petascale machines such as Jaguar and Titan, simulations that can efficiently scale to large node counts are most appropriate. With existing algorithms, approaches that effectively use coarse grain models to achieve sufficient sampling and simulations that require complex materials models are examples. For CPU-based simulations, there has been a trend towards more complicated and accurate potentials for simulations, and it is these models that have the highest potential for relative performance improvements on Titan.

An example is the Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) [17] for simulation of carbon/hydrogen systems. This model can require 55 times longer for force calculation than Lennard-Jones for a similar system size, and efforts already exist for porting this model for GPU acceleration. It is often the case that effective use of accelerators such as GPUs requires the development of new algorithms in order to achieve performance. Because many of the physics models in use today have been derived to optimize well for use on older CPU hardware, the most interesting approaches will likely not arise from porting existing models for GPU acceleration, but rather the development of new models and algorithms that can effectively use fine-grain parallelism and high floating point rates to improve the accuracy and sampling in MD simulations. Simple examples include coarse graining DNA and lipids with aspherical particles using anisotropic potentials such as the Gay-Berne model presented here [18, 19].

5. Acknowledgements

This research was conducted in part under the auspices of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. This research used resources of the Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. M.F.C, J.D.F and P.D.R. acknowledge support from the U.S. Department of Energy, Basic Energy Sciences, Materials Sciences and Engineering Division for supporting the portions of this work related to the design of molecular dynamics computational experiments. The authors thank Yuxing Peng (University of Chicago) for help with long-range partitions. Additionally, we thank Wayne Joubert and the referees for their critical review of the work.

References

[1] Nvidia cuda programming guide version 1.0, nvidia.
[2] Top500 Supercomputer Sites http://www.top500.org (2011).
[3] S. Plimpton, Fast parallel algorithms for short-range molecular-dynamics, Journal of Computational Physics 117 (1) (1995) 1–19.
[4] H. J. D. Baker, J. A., Molecular dynamics simulations using graphics processing units, Molecular Informatics 30 (2011) 498–504.
[5] W. M. Brown, P. Wang, S. J. Plimpton, A. N. Tharrington, Implementing molecular dynamics on hybrid high performance computers - short range forces, Computer Physics Communications 182 (4) (2011) 898–911.
[6] W. M. Brown, A. Kohlmeyer, S. J. Plimpton, A. N. Tharrington, Implementing molecular dynamics on hybrid high performance computers - particle-particle particle-mesh, Computer Physics Communications 183 (3) (2012) 449–459.
[7] M. S. Daw, M. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Physical Review B 29 (12) (1984) 6443–6453.
[8] A. S. Bland, R. A. Kendall, D. B. Kothe, S. G. M. Rogers, J. H., Jaguar: The world's most powerful computer, in: Proceedings of the Cray User Group 2009 Meeting, 2009, pp. 1–7.
[9] M. W. Finnis, J. E. Sinclair, A simple empirical n-body potential for transition metals, Philosophical Magazine A 50 (1) (1984) 45–55.
[10] J. Phillips, G. Zheng, S. Kumar, L. Kale, Namd: Biomolecular simulation on thousands of processors, in: Supercomputing, ACM/IEEE 2002 Conference, 2002, p. 36. doi:10.1109/SC.2002.10019.
[11] W. M. Brown, M. K. Petersen, S. J. Plimpton, G. S. Grest, Liquid crystal nanodroplets in solution, Journal of Chemical Phyics 130 (2009) 044901.
[12] R. Berardi, C. Fava, C. Zannoni, A generalized gay-berne intermolecular potential for biaxial particles, Chem. Phys. Lett. 236 (4-5) (1995) 462–468.
[13] J. D. Fowlkes, L. Kondic, J. Diez, Y. Wu, P. D. Rack, Self-Assembly versus Directed Assembly of Nanoparticles via Pulsed Laser Induced Dewetting of Patterned Metal Films, NANO LETTERS 11 (6) (2011) 2478–2485. doi:10.1021/nl200921c.
[14] M. Fuentes-Cabrera, B. H. Rhodes, J. D. Fowlkes, A. Lopez-Benzanilla, H. Terrones, M. L. Simpson, P. D. Rack, Molecular dynamics study of the dewetting of copper on graphite and graphene: Implications for nanoscale self-assembly, Physical Review E 83 (4, Part 1).
[15] M. Fuentes-Cabrera, B. H. Rhodes, M. I. Baskes, H. Terrones, J. D. Fowlkes, M. L. Simpson, P. D. Rack, Controlling the Velocity of Jumping Nanodroplets Via Their Initial Shape and Temperature, ACS Nano 5 (9) (2011) 7130–7136.
[16] C. J. Fennell, J. D. Gezelter, Is the ewald summation still necessary? pairwise alternatives to the accepted standard for long-range electrostatics, Journal of Chemical Physics 124 (23) (2006) 12.
[17] S. Stuart, A. Tutein, J. Harrison, A reactive potential for hydrocarbons with intermolecular interactions, Journal of Chemical Physics 112 (14) (2000) 6472–6486.
[18] B. Mergell, M. Ejtehadi, R. Everaers, Modeling DNA structure, elasticity, and deformations at the base-pair level, Physical Review E 68 (2, Part 1).
[19] M. Orsi, W. E. Sanderson, J. W. Essex, Permeability of Small Molecules through a Lipid Bilayer: A Multiscale Simulation Study, Journal of Physical Chemistry B 113 (35) (2009) 12019–12029.